ELSEVIER

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Theoretical Computer Science 312 (2004) 75–97

Theoretical Computer Science

www.elsevier.com/locate/tcs

Improving time bounds on maximum generalised flow computations by contracting the network☆

Tomasz Radzik

Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK

Abstract

We consider the maximum generalised network flow problem and a supply-scaling algorithmic framework for this problem. We present three network-modification operations, which may significantly decrease the size of the network when the remaining node supplies become small. We use these three operations in Goldfarb et al.'s supply-scaling algorithm and prove an $\tilde{O}(m^2 n \log B)$ bound on the running time of the resulting algorithm. The previous best time bounds on computing maximum generalised flows are the $O(m^{1.5} n^2 \log B)$ bound of Kapoor and Vaidya's algorithm based on the interior-point method, and the $\tilde{O}(m^3 \log B)$ bound of Goldfarb et al.'s algorithm.

Keywords: Network flows; Generalised flows; Flows with gains and losses

1. Introduction

In a generalised flow network, each arc e has a gain factor $\gamma(e)$ associated with it. If x units of flow enter arc e, then $\gamma(e)x$ units arrive at the other end. Each node has initially a specified amount of *supply* of one common commodity. The objective of the *maximum generalised flow problem* is to design flow which carries these node supplies through the network to one distinguished node, the *sink*. The designed flow must maximise the amount of commodity arriving at the sink and cannot violate the capacities of arcs. This problem models some optimisation problems arising in manufacturing, transportation and financial analysis [1,3,10].

The maximum generalised flow problem is a special case of *linear programming*, so it can be solved by any general-purpose linear programming method. The best asymptotic time bound on computing maximum generalised flows using this approach is

☆ This work was supported by the EPSRC grant GR/L81468.
E-mail address: radzik@dcs.kcl.ac.uk (T. Radzik).

0304-3975/$ - see front matter
doi:10.1016/S0304-3975(03)00403-1

the $O(m^{1.5}n^2 \log B)$ bound of Kapoor and Vaidya's algorithm [9,15] which is based on Karmarkar's interior-point method. Here n is the number of nodes, m is the number of arcs, and B is the largest integer in the representations of the capacities and gain factors of arcs and the supplies at nodes, assuming that these numbers are given as ratios of two integers. Other generalised flow algorithms based on the interior-point method, and matching the time bound of Kapoor and Vaidya's algorithm, were proposed by Murray [10] and Kamath and Palmon [8].

The other line of research in designing generalised-flow algorithms follows the *combinatorial approach* to network flow problems originated by Ford and Fulkerson [2]. A combinatorial algorithm for the maximum generalised flow problem exploits the combinatorial structures of the underlying network and of the flows in this network, and often uses as subroutines combinatorial algorithms for simpler network problems, such as the shortest paths problem and the maximum (non-generalised) flow problem. The first polynomial-time bound on computing maximum generalised flows by a combinatorial algorithm was shown by Goldberg et al. [4], and subsequent improvements are due to Goldfarb and Jin [6], Radzik [12], and Goldfarb, Jin and Orlin [7]. Goldfarb et al.'s $\tilde{O}(m^3 \log B)$ bound [7] has been the best bound prior to our paper. Notation $\tilde{O}(\,)$ hides a factor polylogarithmic in n. The main conclusion of our paper is a combinatorial algorithm which computes maximum generalised flows in $\tilde{O}(m^2 n \log B)$ time. This bound improves Kapoor and Vaidya's bound, if $m = O(n^{2-\varepsilon})$, and Goldfarb et al.'s bound, if $m = \Omega(n^{1+\varepsilon})$, for any constant $\varepsilon > 0$.

Goldfarb et al. [7] presented two algorithms which have the following *supply-scaling* structure. The computation consists of scaling phases. During the current phase, the remaining node supplies are sent in chunks of Δ units towards the sink along the largest gain paths. The scaling parameter Δ decreases at least by half at the end of each phase. Each phase has $O(m)$ iterations and each iteration is dominated by one Dijkstra's single-source shortest-path computation. We present in this paper three network-modification operations, which are intended to decrease the size of the network during the computation of a supply-scaling algorithm. If the network does become smaller, then the subsequent phases may run faster. The first operation is a standard operation of contracting nodes, if there is enough arc capacity between them (in both directions) to accommodate all remaining node supplies. The other two operations are an operation of by-passing (and removing) some nodes and an operation of shortcutting some paths. We believe that in practice these operations may significantly decrease the size of the network and speed-up the computation, but in this paper we focus on the theoretical question of improving the asymptotic worst-case running time.

The computation of Goldfarb et al.'s algorithms [7] terminates when the remaining node supplies total to less than B^{-m}, and an optimal flow is obtained then by simple post-processing. The node supplies drop below B^{-m} in $O(m \log B)$ phases, so the total running time is $\tilde{O}(m^3 \log B)$. We prove that our network-modification operations significantly decrease the size of the network when the remaining node supplies become $B^{-\Omega(n)}$. More precisely, but abstracting from technical details, we show that if the node supplies are $B^{-\Omega(kn)}$, then the number of arcs can be reduced to $O(m/k^2)$. This enables us to decrease the bound on the total running time of all $O(m \log B)$ phases by factor m/n to the new bound of $\tilde{O}(m^2 n \log B)$.

An essential tool in our analysis is a simple fact that the value of an expression composed of additions/subtractions and multiplications/divisions of d fractional numbers with denominators bounded by B is a fractional number with the denominator bounded by B^d. The absolute value of such a number is either 0 or greater than B^{-d}. We use this fact in the following way. If the remaining supply at some node is still positive but less than B^{-d}, then $\Omega(d)$ arcs must "contribute" to the value of this remaining supply. We show a relation between the number of these contributing arcs and the reduction of the size of the network achieved by our network modification operations.

2. Definitions

A (*generalised flow*) *network* $G=(V,E,t,\gamma,u,\delta)$ consists of a set of nodes V, a set of directed arcs E, a *sink*, or *destination*, node $t\in V$, a *gain function* $\gamma: E\to(0,\infty)$, a *capacity function* $u: E\to[0,\infty]$, and a *supply function* $\delta: V\setminus\{t\}\to[0,\infty)$. For an arc $e\in E$, the positive number $\gamma(e)$ is the *gain factor* or the *gain* of arc e. For a node $v\in V\setminus\{t\}$, $\delta(v)$ is the (*initial*) *supply* at node v. The gain factors are sometimes called loss/gain factors.

We normally denote an arc from a node v to a node w by $e_{v,w}$. We allow parallel arcs $e^{(1)}_{v,w}, e^{(2)}_{v,w},\ldots$, but assume that they have different gain factors (parallel arcs cannot be replaced with one arc unless their gain factors are equal). We further assume that all arcs are matched into pairs of *reverse arcs*, and for each pair of reverse arcs $e_{v,w}$ and $e_{w,v}$, $\gamma(e_{w,v})=1/\gamma(e_{v,w})$. Since we allow arc capacities to be zero, the reverse arcs are introduced without loss of generality: if we do not have reverse arcs, then we can add them and set their capacities to zero. The notion of reverse arcs is introduced for notational convenience in the description of residual networks. We denote by n and m the number of nodes and the number of arcs in network G. The arc gain factors and capacities and the node supplies are given as ratios of integers. We denote by B the largest integer among the enumerators and denominators of these input fractional numbers. To simplify asymptotic time bounds, we assume that $B\geqslant n$. Let E_v^- and E_v^+ denote the sets of arcs outgoing from v and incoming to v, respectively. If there is an arc from a node v to a node w, then we call the pair $\{v,w\}$ an *edge*. We denote the set of all edges in network G by $\bar{E}$. The *total supply* $\Delta=\sum_{v\in V\setminus\{t\}}\delta(v)$ is the sum of all node supplies.

For a path or a cycle P in a network G, the gain factor of P is equal to $\gamma(P)=\prod_{e\in P}\gamma(e)$. A cycle P of positive-capacity arcs is called a *flow-generating cycle* or a *flow-absorbing cycle* or a 1-*gain cycle*, depending whether $\gamma(P)$ is greater or less or equal to 1, respectively. If the gain factor of each positive-capacity arc is at most 1, then we call G a *non-gain network*. Clearly, a non-gain network does not have any flow-generating cycle.

A (*generalised*) *flow* f in a network G is a function $f: E\to(-\infty,+\infty)$ which satisfies the following conditions:

(1) *Skew symmetry*: for each pair of reverse arcs e',e'', $f(e'')=-\gamma(e')f(e')$.
(2) *Capacity constraint*: for each arc e, $f(e)\leqslant u(e)$.
(3) *Flow conservation*: for each node $v\neq t$, $\sum_{e\in E_v^-}f(e)\leqslant\delta(v)$.

If $f(e_{v,w})$ units of flow enter arc $e_{v,w}$ at node v, then $\gamma(e_{v,w})f(e_{v,w})$ units arrive at node w. The actual flow (of the underlying commodity) is defined by those flow values $f(e)$ which are positive, while the purpose of the negative flow values on the reverse arcs (Condition 1) is only the notational convenience. For example, using Condition 1, the sum in Condition 3 expresses concisely the *net-flow outgoing* from node v, that is, the actual flow outgoing from v minus the actual flow incoming to v:

$$\sum_{e\in E_v^-, f(e)>0} f(e) - \sum_{e\in E_v^+, f(e)>0} \gamma(e)f(e)$$

$$= \sum_{e\in E_v^-, f(e)>0} f(e) - \sum_{e\in E_v^-, f(e)<0} (-f(e)) = \sum_{e\in E_v^-} f(e).$$

The *value of a flow* f is the net-flow into the sink t. The *maximum generalised network flow problem* is to compute a flow in a given network G of the maximum possible value. Such a flow is called a *maximum flow* or an *optimal flow*. For a flow f in a network G, the *residual capacity* of an arc $e\in E$ and the *residual supply* at a node $v\in V\setminus\{t\}$ are defined as, respectively,

$$u_f(e) = u(e) - f(e), \tag{1}$$

$$\delta_f(v) = \delta(v) - \sum_{e\in E_v^-} f(e) \quad \left(= \delta(v) + \sum_{e\in E_v^+} \gamma(e)f(e)\right). \tag{2}$$

The *residual network* of a network G with respect to a flow f in G is the network $G_f = (V, E, t, \gamma, u_f, \delta_f)$. Note that a network G can be viewed as the residual network G_f with respect to flow $f\equiv 0$. If f' is a flow in network G_f, then $f + f'$ is a flow in network G, and the residual networks $(G_f)_{f'}$ and $G_{f+f'}$ are the same. If f' and f'' are flows in network G, then $f'' - f'$ is a flow in network $G_{f'}$. Thus, an optimal flow f'_{opt} in network G_f gives an optimal flow $f''_{\text{opt}} = f + f'_{\text{opt}}$ in network G. A flow f is a *maximal flow*, if and only if the residual network G_f does not have any flow generating cycle. A *residual arc* is an arc with positive residual capacity. A *residual path* (*cycle*) is a path (cycle) of residual arcs. To simplify the presentation, we consider only networks G such that for any flow f in G, there are residual paths to the sink t from all other nodes. For such networks, the generalised flow optimality conditions, first presented by Onaga [11], can be stated in the following way.

Theorem 1. *A flow f in a network G is a maximum flow, if and only if f is a maximal flow and* $\Delta_f = 0$ (*that is, there are no flow generating cycles and no positive supplies in the residual network* G_f).

The condition that the sink is always reachable from all other nodes along residual paths can be satisfied in the following way. Add to network G n nodes and n arcs forming a path ending at the sink t, and arcs from all nodes in $V\setminus\{t\}$ to the beginning of this path. Setting the capacities of the added arcs to infinity ensures that whatever the flow is, there are always residual paths from all nodes to the sink. Setting the gain

factors of the added arcs to B^{-1} ensures that a maximum flow in the modified network uses these arcs only when there are some remaining node supplies which cannot be sent to the sink along the arcs of the original network G. If we have a maximum flow in the modified network, then the restriction of this flow to the original arcs is a maximum flow in the original network G. Observe also that the modified network has $\Theta(n)$ nodes, $\Theta(m)$ arcs and the parameter B is not changed.

A *labelling* of a network G is a function $\mu : V \to (0, \infty)$ with $\mu(t) = 1$. For a node $v \in V$, $\mu(v)$ is called the *label* of node v. The *re-labelled network* $G_\mu = (V, E, t, \gamma_\mu, u_\mu, \delta_\mu)$ is network G "normalised" with labelling μ:

$$\gamma_\mu(e_{v,w}) = \gamma(e_{v,w}) \frac{\mu(w)}{\mu(v)}, \quad u_\mu(e_{v,w}) = u(e_{v,w})\mu(v), \quad \delta_\mu(v) = \delta(v)\mu(v).$$

If f is a flow in network G, then the same flow expressed in terms of network G_μ is denoted by f_μ, and for each $e_{v,w} \in E$, $f_\mu(e_{v,w}) = f(e_{v,w})\mu(v)$. Note that for $\mu \equiv 1$, $G_\mu \equiv G$. If μ' and μ'' are two labellings of G, then networks $(G_{\mu'})_{\mu''}$ and $G_{\mu'\mu''}$ are the same.

If a network G_f does not have flow generating cycles (that is, if f is a maximal flow in G), then the *canonical labelling* of G_f is defined by the maximum gains of residual paths to the sink. That is, the canonical label $\mu(v)$ is equal to the maximum $\gamma(P)$ over all residual paths P from v to t. The canonical labelling is well defined since we have assumed that there are always residual paths to the sink from all other nodes. If μ is the canonical labelling of network G_f, then we call the re-labelled network $G_{f,\mu}$ a *canonical network*. For each residual arc $e_{v,w}$ in such network $G_{f,\mu}$, $\gamma_\mu(e_{v,w}) \leqslant 1$, because $\mu(v) \geqslant \gamma(e_{v,w})\mu(w) = \gamma_\mu(e_{v,w})\mu(v)$. One can verify that μ is the canonical labelling of G_f, if and only if $G_{f,\mu}$ is a non-gain network and for each node x, there is a 1-gain residual path in $G_{f,\mu}$ from x to t. The canonical labelling of G_f can be computed in $O(nm)$ time by the Bellman–Ford shortest-path algorithm (set the weight of an arc e to $-\log(\gamma(e))$ and compute shortest paths to the sink). If G_f is a non-gain network, then its canonical labelling can be computed in $\tilde{O}(m)$ time using Dijkstra's shortest-paths algorithm (generalised flow algorithms commonly maintain non-gain re-labelled residual networks). The following theorem is proven in [4] (see also [7]).

Theorem 2. *If f is a maximal flow in a network G, μ is the canonical labelling of G_f, and $\Delta_{f,\mu} < B^{-m}$, then a maximum generalised flow in network G can be obtained by one maximum non-generalised flow computation in the network $G_{f,\mu}$ restricted to the arcs with re-label gain γ_μ equal to* 1.

If $G_{f,\mu}$ is a canonical network and $\Delta_{f,\mu} < B^{-m}$, then we call f a *near-optimal flow* in network G. Theorem 2 implies that if we have a near-optimal flow, then we can compute an optimal one in $\tilde{O}(nm)$ additional time needed for the maximum non-generalised flow computation.

For an arbitrary generalised flow network G, one can compute in $\tilde{O}(mn^2 \log B)$ time a maximal flow f_0 by cancelling (saturating) all flow generating cycles [4]. For any

flow f in network G, we have (see (2))

$$\Delta_f = \sum_{v\in V\setminus\{t\}} \delta_f(v) \leqslant \sum_{v\in V\setminus\{t\}} \delta(v) + \sum_{e\in E} \gamma(e)u(e)$$

$$\leqslant nB + mB^2 = \mathrm{O}(mB^2). \tag{3}$$

For any canonical network $G_{f,\mu}$ and any node $v\in V\setminus\{t\}$, $\mu(v)$ is equal to the gain of a simple path, so $B^{-(n-1)}\leqslant\mu(v)\leqslant B^{n-1}$. This and bound (3) imply that $\Delta_{f,\mu}=\mathrm{O}(mB^{n+1})$. In particular, the total supply in the canonical network G_{f_0,μ_0} is $\Delta_{f_0,\mu_0}=\mathrm{O}(mB^{n+1})$, and we will use this bound in the analysis of the running time of generalised-flow algorithms.

If reverse arcs $e_{v,w}$ and $e_{w,v}$ have both positive capacities, then we call them *active arcs* and the edge $\{v,w\}$ an *active edge*. If G is a non-gain network, then the gain factors of active arcs $e_{v,w}$ and $e_{w,v}$ must be both equal to 1 ($\gamma(e_{v,w})\leqslant 1$, $\gamma(e_{w,v})\leqslant 1$ and $\gamma(e_{v,w})\gamma(e_{w,v})=1$). We call a non-gain network G a *basic non-gain network*, if it does not contain a cycle of active edges, does not contain a path of active edges between two nodes with positive supplies, and does not contain a path of active edges between a node with positive supply and the sink. If $G_{f',\mu}$ is a non-gain network, then we can convert the maximal flow f' into another maximal flow f'' such that $G_{f'',\mu}$ is a basic non-gain network. This is done by cancelling cycles of active edges and paths of active edges between a node with positive residual supply and another node with positive residual supply or the sink. Cancelling a cycle of active edges means sending flow along this cycle to saturate at least one arc. If an active cycle has infinite capacity in one direction, then it can be contracted into a single node. Cancelling a path of active edges means sending flow along this path to saturate at least one arc or to remove residual supply from one end node v of the path, $v\neq t$. Such cancelling of cycles and paths can be completed in $\mathrm{O}(nm)$ time using the depth-first search, and in $\mathrm{O}(m\log n)$ time, if the Sleator–Tarjan dynamic-tree data structure [13] is used during the depth-first search. (For an equivalent definition of a basic non-gain network, call a network G_f basic, if and only if f is a vertex of the polytope of flows in G. Then $G_{f,\mu}$ is a basic non-gain network, if and only if it is both a non-gain network and a basic network.)

3. An underlying approximation algorithm

Let ReduceSupply(G, f, μ) be a procedure which takes a flow network G, a maximal flow f in G and the canonical labelling μ of G_f, and computes a maximal flow f' in G and the canonical labelling μ' of $G_{f'}$ such that

$$\Delta_{f',\mu'} \leqslant \Delta_{f,\mu}/2. \tag{4}$$

Algorithm $\mathscr{A}(G)$ shown in Fig. 1 computes a near-optimal flow f in a network G by repeatedly applying procedure ReduceSupply. Let $T(n,m)$ be a bound on the running time of procedure ReduceSupply. The total supply Δ_{f_0,μ_0} in network G_{f_0,μ_0} is $\mathrm{O}(mB^{n+1})$ (see Section 2), and (4) says that each iteration of algorithm $\mathscr{A}$ reduces the

$f \leftarrow$ an initial maximal flow f_0 in G (as discussed in Section 2);
$\mu \leftarrow$ the canonical labeling of G_f;
while $\Delta_{f,\mu} \geq B^{-m}$ **do** $(f, \mu) \leftarrow \text{ReduceSupply}(G, f, \mu)$.

Fig. 1. Algorithm $\mathscr{A}(G)$. Output: a near optimal flow f in network G.

total supply $\Delta_{f,\mu}$ at least by half. Hence, algorithm $\mathscr{A}$ computes a near-optimal flow in $O(\log(mB^{n+1}/B^{-m})) = O(m \log B)$ iterations, or in $\tilde{O}(mn^2 \log B) + O(T(n,m)m \log B)$ total time (the first term is the bound on the running time of the computation of the initial maximal flow f_0). Goldfarb et al. [7] introduced this algorithmic framework $\mathscr{A}$ and showed two different procedures ReduceSupply. Each of them consists of $O(m)$ iterations, and each iteration is dominated by Dijkstra's shortest-path computation, so $T(n,m) = \tilde{O}(m^2)$. Thus Goldfarb et al. [7] showed that a near-optimal flow can be computed in $\tilde{O}(m^3 \log B)$ time (and a maximum generalised flow can be computed within the same time bound; see Theorem 2). We prove in this paper the following main theorem.

Theorem 3. *A near-optimal flow in network G can be computed in $\tilde{O}(mn^2 \log B + m^2) + O(T(n,m)n \log B)$ time, assuming that the bounding function $T(n,m)$ increases in the following way*:

$$T(n_1, cm) \leqslant cT(n_2, m) \quad \text{if } n_1 \leqslant n_2 \text{ and } c \leqslant 1. \tag{5}$$

Theorem 3 implies that using Goldfarb et al.'s ReduceSupply procedures, near-optimal flows can be computed in $\tilde{O}(m^2 n \log B)$ time, so the main conclusion of our paper is the following theorem.

Theorem 4. *A maximum generalised flow can be computed in $\tilde{O}(m^2 n \log B)$ time.*

We obtain an algorithm $\mathscr{A}'$ with the running time as claimed in Theorem 3 by modifying algorithm $\mathscr{A}$. The idea is to keep reducing the size of the intermediate canonical networks $G_{f,\mu}$ by periodically applying three operations described later in Sections 4.2–4.4. The input to these operations is a non-gain or a basic non-gain network G, depending on the operation, and the output is a non-gain network H. The set of nodes of H is a subset of the set of nodes of G, but the set of arcs of H is not necessary a subset of the set of arcs of G. For the correctness of algorithm $\mathscr{A}'$ and the analysis of its running time, the following properties of our network-modification operations will be important.

(1) The input and the output networks G and H have the same node supplies (that is, each node which is in both G and H has the same supply in both networks and each node which is in G but not in H has zero supply in G).
(2) If the input network G is a basic canonical network, then so is the output network H.

(3) If f' is a maximal flow in network H and μ' is the canonical labelling of $H_{f'}$, then there is a simple way of obtaining a maximal flow f in network G and the canonical labelling μ of G_f such that the canonical networks $H_{f',\mu'}$ and $G_{f,\mu}$ have the same node supplies.

4. Reducing the size of network

4.1. Large-capacity arcs

For a non-gain network G, we say that an arc e has *large capacity* if $u(e) > \Delta$, and *small capacity* otherwise. The following lemma implies that large capacities can be assumed to be infinity.

Lemma 5. *If G is a non-gain network and e is a large-capacity arc in G, then for every flow f in G which is not positive on any directed cycle containing arc e, $f(e) < u(e)$.*

The validity of Lemma 5 comes from the following intuition. The flow $f(e)$ on arc e must come from the node supplies and since the gain factors of positive-capacity arcs are at most 1, these supplies can only diminish while passing through the network. Lemma 5 can be easily proven by a formalisation of this intuition using a decomposition theorem for generalised flows (see [4,12] for such theorems). If G' is a non-gain network obtain from a non-gain network G by changing a large capacity of an arc to infinity and f' is a maximal flow in G', then the removal of flow from 1-gain cycles gives a maximal flow f in G. In particular, a maximum flow in G' gives in this way a maximum flow in G. We describe in the next section and in the following three subsections how we use the notion of large capacities to modify a network in order to reduce its size.

Let G be a non-gain network. If there are two or more large-capacity parallel arcs $e^{(1)}_{v,w}, e^{(2)}_{v,w}, \ldots$ in G, then the lemma below implies that each maximal flow in G must be equal to 0 on all these arcs except possibly on the one with the largest gain factor. Thus, we can remove all these arcs (and their reverse arcs) except the one with the largest gain factor.

Lemma 6. *Let G be a non-gain network with two large-capacity arcs $e^{(1)}_{v,w}$ and $e^{(2)}_{v,w}$, and let $\gamma(e^{(1)}_{v,w}) > \gamma(e^{(2)}_{v,w})$. For each maximal flow f in G, $f(e^{(2)}_{v,w}) = 0$.*

Proof. Let $e^{(2)}_{w,v}$ be the reverse arc to $e^{(2)}_{v,w}$. We have $1/\gamma(e^{(2)}_{w,v}) = \gamma(e^{(2)}_{v,w}) < \gamma(e^{(1)}_{v,w}) \leqslant 1$, so $\gamma(e^{(2)}_{w,v}) > 1$, which implies that $u(e^{(2)}_{w,v}) = 0$ (in a non-gain network, the gain factors of positive capacity arcs are at most 1), so $f(e^{(2)}_{v,w}) \geqslant 0$. We assume that $f(e^{(2)}_{v,w}) > 0$, and show that this implies that flow f is not maximal. Since $f(e^{(2)}_{v,w}) > 0$, $e^{(2)}_{w,v}$ is a residual arc in G_f. If $f(e^{(1)}_{w,v}) < u(e^{(1)}_{w,v})$, then arcs $e^{(1)}_{w,v}$ and $e^{(2)}_{w,v}$ form a flow-generating cycle in G_f, so flow f is not maximal. If $f(e^{(1)}_{v,w}) = u(e^{(1)}_{v,w})$, then Lemma 5 implies that f is

positive on a cycle Γ containing $e^{(1)}_{v,w}$. The cycle reverse to Γ with arc $e^{(1)}_{w,v}$ replaced with arc $e^{(2)}_{w,v}$ is a flow-generating cycle in G_f (the gain of each arc on this cycle is at least 1 and the gain of arc $e^{(2)}_{w,v}$ is greater than 1), so flow f is not maximal.

4.2. Contracting large-capacity edges

Let G be a non-gain network. If the capacities of a pair of reverse arcs $e_{v,w}$ and $e_{w,v}$ are both large, then we call edge $\{v,w\}$ a *large-capacity edge* or a *contractable edge*. Observe that arcs $e_{v,w}$ and $e_{w,v}$ are active, so their gain factors must be equal to 1. To contract a large-capacity edge $\{v,w\}$, where $w \neq t$ and $\delta(w) = 0$, means to create a new network H by modifying network G in the following way. For each arc adjacent to node w, replace the end node w with node v. Then remove node w from the network. The arc gains, arc capacities and node supplies in H are the same as in G.

For a basic non-gain network G, let $\bar{F}^a \subseteq \bar{E}$ and $\bar{F}^c \subseteq \bar{F}^a$ denote the set of active edges and the set of contractable edges, respectively. By definition, a basic network does not have a cycle of active edges, so the undirected graphs $(V, \bar{F}^a)$ and $(V, \bar{F}^c)$ are forests. A *component* $C \subseteq V$ of network G is a connected component of forest $(V, \bar{F}^a)$, and a *strong component* $C \subseteq V$ is a connected component of forest $(V, \bar{F}^c)$. A component of G contains at most one node which has positive supply or is the sink t (this is another consequence of the definition of a basic network). Let H be the basic non-gain network obtained from network G by contracting all contractable edges (all edges in $\bar{F}^c$), removing loop arcs, and removing all arcs but one from each set of parallel large-capacity arcs (as described in Section 4.1). We will refer to this whole three-part operation as *contracting all contractable edges*. This operation can be implemented so that its running time is $O(m)$. A strong component C of network G contracts to one node $r \in C$. We say that this node r represents strong component C in network H. If $t \in C$, then $r = t$, and if C contains a node with positive supply, then r is this node. Otherwise r is an arbitrary node in C. The node supplies in network H are clearly the same as in network G, and if G is a basic canonical network, then so is H.

Let $\hat{G}$ be the network G with the capacities of the arcs of all contractable edges changed to infinity. If f' is a maximal flow in network H, then in $O(n)$ time we can extend f' to a maximal flow $\hat{f}$ in network $\hat{G}$ by connecting flow along the arcs of the edges in $\bar{F}^c$. We obtain the canonical labelling μ of network $\hat{G}_{\hat{f}}$ from the canonical labelling μ' of network $H_{f'}$ by setting $\mu(v)$ to the label μ' of the node in H which represents the strong component of G containing v. The canonical networks $H_{f',\mu'}$ and $\hat{G}_{\hat{f},\mu}$ have the same node supplies. Modify flow $\hat{f}$ by removing flow from directed cycles to obtain a maximal flow f in G. The canonical networks $G_{f,\mu}$ and $H_{f',\mu'}$ have the same node supplies.

4.3. By-passing and removing free nodes

The operation of contracting all contractable edges may not succeed on its own in reducing the size of the network. Even if we have a residual network with a very small remaining total supply (so with many large-capacity arcs), we may still have

only relatively few contractable edges. In this subsection, we describe our second network modification operation applicable to a non-gain network G. We say that a node $v \in V \setminus \{t\}$ is *free*, if it does not have positive supply ($\delta(v) = 0$) and for each pair of reverse arcs adjacent to v, the capacity of one of them is large and the capacity of the other one is zero. Let $V_{\text{free}} \cup V_{\text{non-free}} = V$ be the partitioning of the set of nodes into the free nodes and the non-free nodes. A free node $v \in V_{\text{free}}$ can be removed from the network in the following way. For each pair of positive (hence large) capacity arcs $e_{x,v}$ and $e_{v,z}$, add a new arc $e_{x,z}$ with infinite capacity and gain equal to $\gamma(e_{x,v})\gamma(e_{v,z})$, and add the zero-capacity arc $e_{z,x}$ reverse to arc $e_{x,z}$. For each set of parallel large-capacity arcs, remove all of them except the one with the largest gain factor. Then remove from the network node v and all arcs adjacent to it. For each flow in the obtained non-gain network, there is a naturally corresponding flow in the original network G (re-route the flow from the added arc $e_{x,z}$ onto arcs $e_{x,v}$ and $e_{v,z}$).

The removal of all free nodes by iteratively applying the above procedure is equivalent to the following computation. For each pair of non-free nodes x and z, add a new infinite-capacity arc $e_{x,z}$ and a zero-capacity reverse arc $e_{z,x}$. The gain factor of the added arc $e_{x,z}$ is equal to the largest gain of a path in G from x to z which passes only through free nodes. If there is no such path, then do not introduce arc $e_{x,z}$. The gain factors of all new arcs can be computed in $\tilde{O}(mn')$ time, where $n' = |V_{\text{non-free}}|$, by applying Dijkstra's single-destination shortest-paths algorithm to each non-free node as the destination. Then remove all free nodes, their adjacent arcs, and all arcs but one from each set of parallel large-capacity arcs. We call this whole computation the operation of removal of all free nodes. Let H denote the non-gain network obtained from a non-gain network G by this operation. The node supplies are clearly the same in H as in G, and if G is a basic canonical network, then so is H. If we have stored the n' largest-gain-path trees computed to set the gain factors of the new arcs, and we have a maximal flow f' in network H, then we can compute a maximal flow f in network G in $O(nn') = O(n^2)$ time in the following way. For each node z in H, move the flow from the added arcs $e_{x,z}$ onto the arcs of the largest-gain-path tree rooted at z. Update the flow on the arcs of the tree starting from the leaves for the running time of $O(n)$ per one tree. Finally, remove flow from directed cycles to ensure that no large capacity in network G is violated. For the nodes in $V_{\text{non-free}}$, the canonical labelling μ of G_f is the same as the canonical labelling μ' of $H_{f'}$. The labels μ of the nodes in V_{free} can be computed in $\tilde{O}(m)$ time using Dijkstra's shortest-paths algorithm. Clearly, the node supplies in networks $H_{f',\mu'}$ and $G_{f,\mu}$ are the same.

4.4. Shortcutting small reverse-flow paths

In this subsection, we first give an example of an obstacle which may make the operations of the previous two subsections ineffective, and then propose our third network modification operation, which overcomes such obstacles.

Let G be a network with a long directed path R such that all arcs reverse to the arcs on this path have zero capacity. Let f be a maximal flow in G and μ be the canonical labelling of G_f such that $f_\mu(e)$ is the same small positive value for each arc e on path R. In network $G_{f,\mu}$, all edges of path R are non-contractable (the arcs reverse to

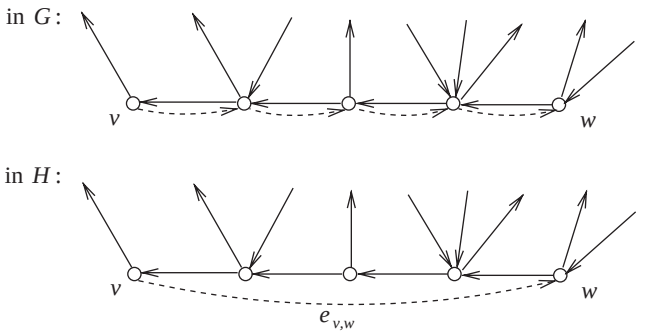

Fig. 2. Shortcutting a small reverse-flow path P from v to w (dash arcs). Only the positive-capacity arcs adjacent to the path are shown. The arcs on the path have the same, small capacity, while the other arcs have large capacities.

the arcs on R have small residual capacities) and clearly none of the nodes on path R is free. Thus, if such a case persists throughout the computation of a maximum generalised flow algorithm $\mathscr{A}$, then the operations introduced in Sections 4.2 and 4.3 may not help in reducing the size of the network. Next we describe an operation of "shortcutting" such paths, which may make the intermediate nodes of the path free. Our description does not refer explicitly to a flow and a residual network, but in generalised-flow algorithms this operation would be applied to residual networks.

Let G be a non-gain network, and let P be a directed path from a node v to a node w of length at least two and such that the capacities of all arcs on the path are the same, small and positive, and the capacities of all arcs reverse to the arcs on P are large (see Fig. 2). We call such a path P a *small reverse-flow path* because it would occur as the reverse residual path of a small flow from node w to node v. Observe that the arcs on P are active, so their gain factors are equal to 1. (If network G here is the canonical residual network $G_{f,\mu}$ from the example given in the previous paragraph, then path P is the path reverse to path R.) We modify network G by adding a new arc $e_{v,w}$ with the capacity equal to the capacity of path P and with the gain factor equal to 1. We also add a zero capacity reverse arc $e_{w,v}$ and set the capacities of the arcs on P to zero. Observe that if there are no positive supplies at the intermediate nodes on P and all positive-capacity arcs adjacent to these nodes other than the arcs of P have large capacities, then this shortcutting operation makes free all intermediate nodes on P other than the sink (if the sink is on P). For us this will be precisely the purpose of using this operation: make more free nodes and then remove all of them.

The obtained network H is a non-gain network with the same node supplies as in network G, and if G is a basic canonical network, then so is H. Let f' be a maximal flow in network H, and let μ be the canonical labelling of $H_{f'}$. Let $\hat{f}$ be a maximal flow in H obtained from flow f' by removal of flow from directed cycles. Let f be the following flow in G: $f(e)=\hat{f}(e)+\hat{f}(e_{v,w})$ and $f(e')=\hat{f}(e')+\hat{f}(e_{w,v})$, for each arc e on path P and arc e' reverse to e, and $f(e)=\hat{f}(e)$, for all other arcs in G. Networks $H_{f',\mu}$, $H_{\hat{f},\mu}$ and $G_{f,\mu}$ have the same node supplies. The following lemma implies that f is a maximal flow in G and μ is the canonical labelling of G_f.

Lemma 7. *There is a residual path in $H_{\hat{f}}$ from a node p to a node q with gain g, if and only if, there is a residual path in G_f from p to q with gain g.*

Proof. The arcs reverse to the arcs of path P are residual in both networks $H_{\hat{f}}$ and G_f, because their capacities are large. Thus, the only differences between networks $H_{\hat{f}}$ and G_f are of the following two types. One or both of the arcs $e_{v,w}$ and $e_{w,v}$ are residual in $H_{\hat{f}}$, while they are not present in G_f; and there may be arcs of path P which are residual in G_f but not in $H_{\hat{f}}$.

Let Q be a residual path from p to q in $H_{\hat{f}}$. If Q does not contain arc $e_{v,w}$ or arc $e_{w,v}$, then it must be also a residual path in G_f. If Q contains arc $e_{v,w}$, then path Q' obtained from path Q by replacing this arc with path P is a residual path from p to q in G_f (if $e_{v,w}$ is residual in $H_{\hat{f}}$, then all arcs of P are residual in G_f). Paths Q and Q' have the same gain. If Q contains arc $e_{w,v}$, then path Q' obtained from Q by replacing this arc with the path reverse to path P is a residual path from p to q in G_f, and paths Q and Q' have the same gain.

Now let Q be a residual path from p to q in G_f. If Q does not contain an arc from P which is not residual in $H_{\hat{f}}$, then Q is also a residual path in $H_{\hat{f}}$. Assume therefore that Q contains an arc $e_{x,y}$ from P which is not residual in $H_{\hat{f}}$. For each such arc, $\hat{f}(e_{y,x}) = 0$ (otherwise $e_{x,y}$ would be residual in $H_{\hat{f}}$) and $\hat{f}(e_{v,w}) < u(e_{v,w})$ (if $\hat{f}(e_{y,x}) = 0$ and $\hat{f}(e_{v,w}) = u(e_{v,w})$, then $e_{x,y}$ would not be residual in G_f). Let Γ denote the residual cycle in $H_{\hat{f}}$ formed by arc $e_{v,w}$ and the path reverse to path P. Let Q' be the path obtained from path Q by replacing each arc $e_{x,y}$ from $Q \cup P$ with the path $\Gamma - \{e_{y,x}\}$. Q' is a residual path from p to q and has the same gain as path Q. □

An arc e on a small reverse-flow path and the arc reverse to it form an active edge. If G is a basic non-gain network, then the active edges form a forest $\bar{F}^a$. We identify in $O(m)$ time all active arcs and then we identify in $O(n)$ time all maximal (that is, non-extendible) small reverse-flow paths such that the intermediate nodes have degree two in forest $\bar{F}^a$ (observe that these paths are pairwise arc disjoint and uniquely defined). By *shortcutting small reverse-flow paths* in a basic non-gain network G we mean shortcutting all these maximal small reverse-flow paths. The running time of this operation is $O(m)$.

4.5. Putting it all together: procedure SHRINK

Let G be a basic non-gain network. Procedure SHRINK(G) modifies network G by executing the following steps:

(1) Contract all contractable edges in G to obtain a basic non-gain network $G^{(1)}$ (the running time of $O(m)$).
(2) Shortcut small reverse-flow paths in the basic non-gain network $G^{(1)}$ to obtain a basic non-gain network $G^{(2)}$ (the running time of $O(m)$).
(3) Remove all free nodes from network $G^{(2)}$ to obtain a basic non-gain network $G^{(3)}$ (the running time of $\tilde{O}(mn)$).
(4) Return network $G^{(3)}$ or the initial network G, whichever has fewer arcs.

The $\tilde{O}(nm)$ bound on the running time of step (3) is also a bound on the running time of the whole procedure. The node supplies in networks G and $G^{(3)}$ are the same, and $G^{(3)}$ is a basic non-gain network. If G is a basic canonical network, then so is $G^{(3)}$. If we have a maximal flow f' in $G^{(3)}$ and the canonical labelling μ' of $G^{(3)}_{f'}$, then we can compute in $O(n^2+m)$ time a maximal flow f in G and the canonical labelling μ of G_f such that networks $G_{f,\mu}$ and $G^{(3)}_{f',\mu'}$ have the same node supplies. Fig. 3 illustrates procedure SHRINK$(G_{f,\mu})$, where $G_{f,\mu}$ is a basic canonical network with small residual supply.

We modify algorithm $\mathscr{A}(G)$ described in Section 3 into the algorithm $\mathscr{A}'(G)$ shown in Fig. 4. Algorithm $\mathscr{A}'$ maintains a maximal flow f in network G and the canonical labelling μ of network G_f. One *stage* of the algorithm—one iteration of the outer loop—consists of the following computation. We first modify the current maximal flow f so that $G_{f,\mu}$ becomes a basic canonical network. Then we apply procedure SHRINK to network $G_{f,\mu}$ and denote by H the obtained basic canonical network. Next we apply q times procedure REDUCESUPPLY to network H and obtain a maximal flow f' in H and the canonical labelling μ' of $H_{f'}$. From flow f' and labelling μ' in H we compute a maximal flow $\bar{f}$ in G_f and the canonical labelling μ^{new} of $G_{f^{\text{new}}}$, where $f^{\text{new}}=f+\bar{f}$ ($\mu^{\text{new}}=\mu\bar{\mu}$, where $\bar{\mu}$ is the canonical labelling of $G_{f^{\text{new}},\mu}$ derived from the canonical labelling μ' of $H_{f'}$). The new flow and labelling in network G at the end of the stage are f^{new} and μ^{new}. At the beginning of the stage the node supplies in networks $G_{f,\mu}$ and H are the same, and at the end of the stage the node supplies in networks $H_{f',\mu'}$ and $G_{f^{\text{new}},\mu^{\text{new}}}$ are the same. Hence, one stage of algorithm $\mathscr{A}'$ reduces the total supply $\Delta_{G,f,\mu}$ at least by factor 2^{-q} (one stage of algorithm $\mathscr{A}'$ corresponds to q iterations of algorithm $\mathscr{A}$).

To show a better asymptotic bound on the running time of algorithm $\mathscr{A}'$ than the bound for algorithm $\mathscr{A}$, we have to know when procedure SHRINK provably reduces the size of the network. The following theorem says that the size of the network does decrease, if the total residual supply is exponentially small.

Theorem 8. *There exists a constant $\alpha>0$ such that if $G_{f,\mu}$ is a basic canonical network, $\Delta_{f,\mu}\leqslant B^{-3kn}$, and $1\leqslant k\leqslant m/n$, then procedure* SHRINK$(G_{f,\mu})$ *returns a canonical network with at most n nodes and at most* $\min\{m, 2n^2, [(\alpha m)/(kn)]^2\}$ *arcs.*

Using Theorem 8, we now prove a bound on the running time of algorithm $\mathscr{A}'$, which immediately implies Theorem 3. We prove Theorem 8 in Sections 5 and 6.

Theorem 9. *There exists a constant c such that if $q=\lceil cn\log B\rceil$ and the bound $T(n,m)$ on the running time of procedure* REDUCESUPPLY *satisfies* (5)*, then the running time of algorithm $\mathscr{A}'(G)$, excluding the computation of the initial maximal flow f_0 and the labelling μ_0, is* $O(T(n,m)n\log B)+\tilde{O}(m^2)$.

Proof. We number the stages of algorithm $\mathscr{A}'$ from 0 and denote by f_k and μ_k the maximal flow in G and the canonical labelling of G_{f_k} at the beginning of stage k. For

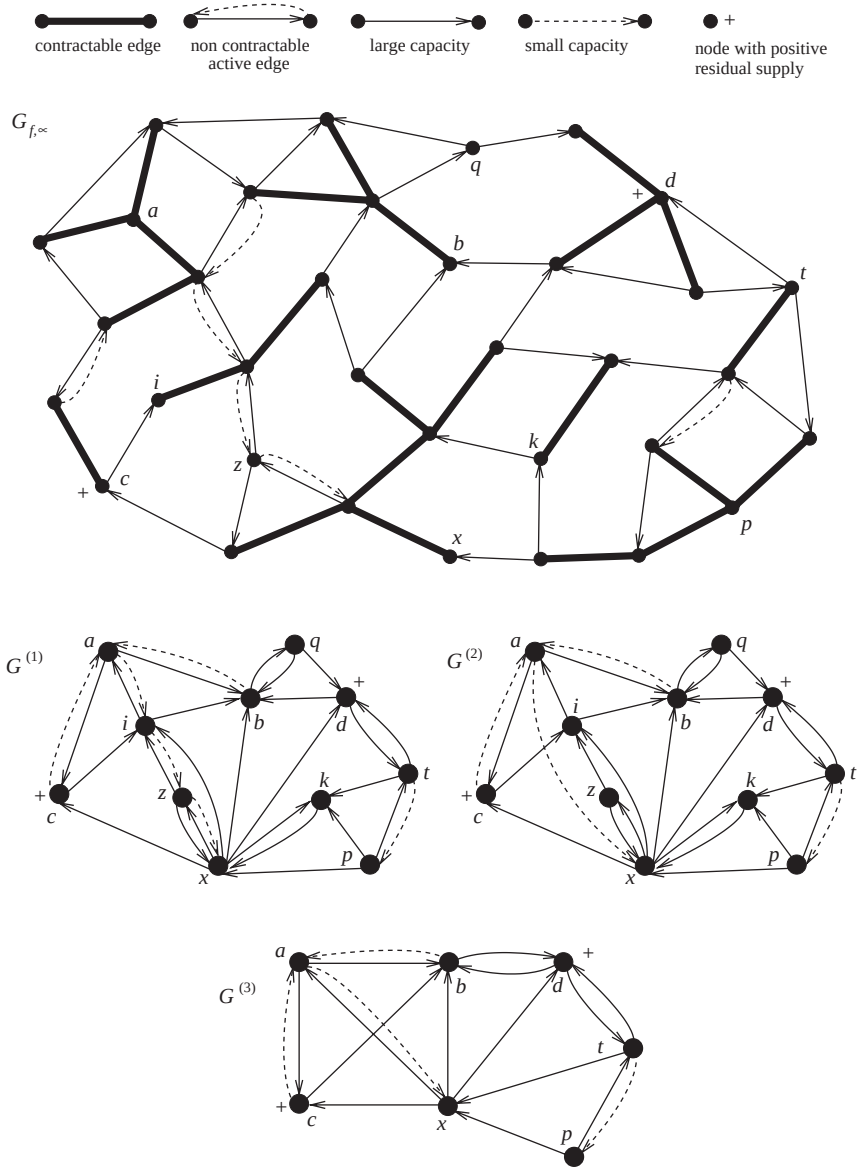

Fig. 3. Illustration of procedure SHRINK($G_{f,\mu}$), where $G_{f,\mu}$ is a basic canonical network with small residual supply. Zero-capacity arcs are not shown. Each non-contractable active edge has one small-capacity arc and one large capacity arc (see Lemma 14). Path $\langle a, i, z, x\rangle$ in $G^{(1)}$ is shortcut in $G^{(2)}$. The free nodes in $G^{(2)}$ are: i, z, q and k.

each stage $k \geqslant 0$,

$$\Delta_{f_{k+1},\mu_{k+1}} \leqslant 2^{-q}\Delta_{f_k,\mu_k} \leqslant B^{-cn}\Delta_{f_k,\mu_k}. \tag{6}$$

$(f,\mu) \leftarrow$ an initial maximal flow in G and the canonical labeling of G_{f_0};
while $\Delta_{f,\mu} \geq B^{-m}$ **do**
  $f \leftarrow$ modified flow f so that $G_{f,\mu}$ is a basic canonical network;
  $H \leftarrow$ Shrink($G_{f,\mu}$);
  $(f',\mu') \leftarrow$ zero flow and unit labeling in H ($f' \equiv 0$, $\mu' \equiv 1$);
  repeat q times: $(f',\mu') \leftarrow$ ReduceSupply(H, f', μ');
  $\bar{f} \leftarrow$ maximal flow in G_f derived from f' in H;
  $f^{\text{new}} \leftarrow f + \bar{f}$; $\mu^{\text{new}} \leftarrow$ the canonical labeling of $G_{f^{\text{new}}}$;
  $(f,\mu) \leftarrow (f^{\text{new}}, \mu^{\text{new}})$;
end_while.

Fig. 4. Algorithm $\mathscr{A}'(G)$. Output: a near-optimal flow f in G.

The total supply Δ_{f_0,μ_0} at the beginning of stage 0 is $O(mB^{n+1})$, so (6) implies that for a suitably large constant c and for $k \geqslant 1$,

$$\Delta_{f_k,\mu_k} \leqslant B^{-cnk}\Delta_{f_0,\mu_0} = B^{-cnk}O(mB^{n+1}) \leqslant B^{-3kn}. \tag{7}$$

Inequality (7) implies that the total residual supply $\Delta_{f,\mu}$ drops below the B^{-m} threshold by the beginning of stage $K = \lceil m/(3n) \rceil$. Let n_k and m_k be the numbers of nodes and arcs in the network computed by procedure SHRINK(G_{f_k,μ_k}), that is, in the network used during stage k. Theorem 8 implies that $n_k \leqslant n$ and

$$m_k \leqslant \min\left\{m, 2n^2, \frac{(\alpha m)^2}{(kn)^2}\right\}. \tag{8}$$

The running time of stage k is at most $T(n_k,m_k)\lceil cn \log B \rceil$, for the $\lceil cn \log B \rceil$ applications of procedure REDUCESUPPLY, plus $\tilde{O}(nm)$ for all other computation. Thus, the running time of algorithm $\mathscr{A}'$ is at most $K \cdot \tilde{O}(nm) = \tilde{O}(m^2)$ plus $\lceil cn \log B \rceil$ times

$$\sum_{k=0}^{K-1} T(n_k,m_k) = \sum_{k=0}^{K-1} T\left(n_k, \frac{m_k}{m}m\right) \leqslant \frac{T(n,m)}{m}\sum_{k=0}^{K-1} m_k. \tag{9}$$

The inequality in (9) follows from property (5) of function $T(n,m)$. We show that $\sum_{k=1}^{K-1} m_k = O(m)$ (note that $m_0 \leqslant m$). Let $K_0 = \lceil \alpha m/n^2 \rceil$ (there may be parallel arcs in G so we may have $m > n^2$). If $K_0 > 1$, then $K_0 \leqslant 2\alpha m/n^2$ and, using the second bound of (8), we have

$$\sum_{k=1}^{K_0-1} m_k \leqslant 2n^2K_0 \leqslant 4\alpha m = O(m).$$

We bound the sum $\sum_{k=K_0}^{K-1} m_k$ using the third bound of (8) and the fact that $\sum_{i=p}^{\infty} 1/i^2 = \Theta(1/p)$.

$$\sum_{k=K_0}^{K-1} m_k \leqslant \sum_{k=K_0}^{K-1} \frac{(\alpha m)^2}{(kn)^2} = \frac{(\alpha m)^2}{n^2}\sum_{k=K_0}^{K-1}\frac{1}{k^2} \leqslant \alpha m K_0 \sum_{k=K_0}^{\infty}\frac{1}{k^2} = O(m).$$

This concludes the proof of Theorem 9. □

5. Size of strong components when residual supply is very small

In this section, we prove lemmas which we need for proving Theorem 8. For a set of nodes $C \subseteq V$, define its degree $\deg(C)$ as the number of arcs with at least one end in C. The proof of Theorem 8 is based on Lemmas 11–13, which say that when the total residual supply is exponentially small, then a strong component of a certain type must have large degree. These three lemmas consider three different types of strong components. The reason for looking at the degrees of strong components is that if the strong components have large degrees, then there cannot be too many of them, so the network computed by procedure SHRINK cannot have too many nodes (the details of this argument are in Section 6). The proofs of Lemmas 11–13 are based on Lemma 10, which gives an expression on the balance of flow at a subset of nodes, and on the following observation. To obtain a positive number less then B^{-d} by applying additions/subtractions and multiplications/divisions to the fractional input numbers, we have to use at least d of those numbers. We also need Lemma 14, which says that when the total residual supply is very small, at least one arc in each pair of reverse arcs must have large residual capacity. A *bidirectional tree* $T \subseteq E$ spanning a set of nodes $C \subseteq V$, $C \neq \emptyset$, consists of $|C| - 1$ pairs of reverse arcs spanning C.

Lemma 10. *Let T be a bidirectional tree in a network G spanning a non-empty set of nodes $C \subseteq V$. Let r be an arbitrary node in C, and for a node $v \in C$, let $P_v \subseteq T$ denote the tree path from v to the root r. If f is a flow in network G, then the following equation holds*:

$$\sum_{v \in C} \delta_f(v)\,\gamma(P_v) - \sum_{v \in C} \delta(v)\gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v^- \setminus T} f(e)\gamma(P_v) = 0. \tag{10}$$

Proof. From the definition of the residual supplies (2), for each $v \in V$,

$$\delta_f(v) - \delta(v) + \sum_{e \in E_v^-} f(e) = 0. \tag{11}$$

For each $v \in C$, multiply (11) by $\gamma(P_v)$ and add all equations together to get

$$\sum_{v \in C} \delta_f(v)\gamma(P_v) - \sum_{v \in C} \delta(v)\gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v^-} f(e)\gamma(P_v) = 0.$$

In the double sum above, take aside the terms corresponding to the arcs in T:

$$\sum_{v \in C} \sum_{e \in E_v^-} f(e)\gamma(P_v) = \sum_{v \in C} \sum_{e \in E_v^- \setminus T} f(e)\gamma(P_v) + \sum_{e_{x,y} \in T} f(e_{x,y})\gamma(P_x).$$

Conclude the proof by observing that the last sum above is equal to zero. Indeed, by the skew symmetry of flows, the pair of terms in this sum corresponding to a pair of reverse arcs $e_{x,y}$ and $e_{y,x}$ in T cancels out:

$$f(e_{x,y})\gamma(P_x) = -f(e_{y,x})\gamma(e_{y,x})\gamma(P_x) = -f(e_{y,x})\gamma(P_y).$$

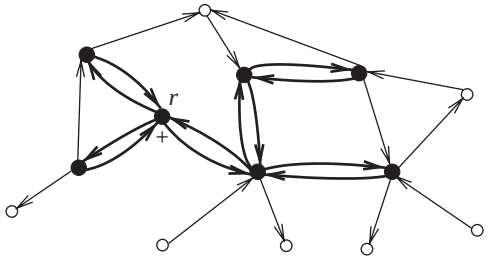

Fig. 5. A strong component of Lemma 11. Black nodes are in C, white nodes are adjacent to C. Only positive residual-capacity arcs are shown; the tree arcs in bold.

Lemma 11. *Let $G_{f,\mu}$ be a basic canonical network and let $C \subseteq V \setminus \{t\}$ be a strong component of $G_{f,\mu}$ with the following properties. There are no active non-contractable edges adjacent to C (that is, C is also a component) and there is a positive supply at one node in C. Under these conditions, if an integer d is such that $d \geqslant n$ and $\Delta_{f,\mu} \leqslant B^{-3d}$, then the degree of set C is at least d.*

Proof. Let r be the node in C which has positive supply (by definition, a component may have at most one such node) and let T be the bidirectional tree of active arcs spanning C (see Fig. 5). Eq. (10) becomes here

$$\delta_f(r) - \sum_{v \in C} \delta(v)\gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v^- \setminus T} f(e)\gamma(P_v) = 0. \tag{12}$$

We assume that the degree of C is at most $d-1$ and derive a contradiction. All arcs in the double sum in (12) are non-active because C is a component. Hence for each arc e in this sum, $f(e)$ is either at its upper bound $u(e)$, or at its lower bound $-u(e')\gamma(e')$, where e' is the reverse arc to arc e. Let D be the (multi-)set of the denominators of the input numbers involved in (12):

(1) numbers $\delta(v)$, for each $v \in C$ ($|C|$ numbers);
(2) either number $u(e)$, or numbers $u(e')$ and $\gamma(e')$, for each non-tree arc e with the tail in C, where e' is the reverse arc to arc e (at most $2(\deg(C) - 2(|C| - 1))$ numbers);
(3) and numbers $\gamma(e)$, for each arc $e \in T$ which is in the direction towards the root node r ($|C| - 1$ numbers).

Thus, there are at most $2\deg(C) + 1 \leqslant 2d - 1$ numbers in D and each of them is at most B. Hence, the absolute value of the left-hand side of (12) excluding $\delta_f(r)$ is a fractional number with denominator at most B^{2d-1}, so it is either equal to 0 or greater than B^{-2d}. This contradicts Eq. (12) because the positive number $\delta_f(r)$ is less than B^{-2d}:

$$0 < \delta_f(r) = \delta_{f,\mu}(r)/\mu(r) \leqslant \Delta_{f,\mu} B^{n-1} < B^{-3d+n} \leqslant B^{-2d}.$$

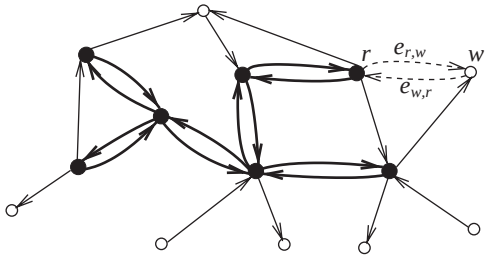

Fig. 6. Strong component considered in Lemma 12.

The second inequality above follows from the fact that if $G_{f,\mu}$ is a canonical network, then $B^{-(n-1)} \leqslant \mu(v) \leqslant B^{n-1}$ for each node v. □

Lemma 12. *Let $G_{f,\mu}$ be a basic canonical network and let $C \in V \setminus \{t\}$ be a strong component of network $G_{f,\mu}$ with the following properties. There is exactly one active but non-contractable edge adjacent to C and there is no residual supply in C. Under these conditions, if an integer d is such that $d \geqslant n$ and $\Delta_{f,\mu} \leqslant B^{-3d}$, then the degree of set C is at least d.*

Proof. Let T be the bidirectional tree of active arcs spanning C. Let $e_{r,w}$ and $e_{w,r}$ be the pair of active arcs forming that unique active but non-contractable edge adjacent to C, and let $r \in C$ (see Fig. 6). There are no residual supplies in C, so Eq. (10) of Lemma 10 becomes in this case

$$-\sum_{v \in C} \delta(v)\gamma(P_v) + \sum_{v \in C} \sum_{e \in E_v^- \setminus (T \cup \{e_{r,w}\})} f(e)\gamma(P_v) + f(e_{r,w}) = 0. \tag{13}$$

We assume that the degree of set C is at most $d-1$ and derive a contradiction in a similar way as in the proof of Lemma 11. All non-tree arcs with tails in C except arc $e_{r,w}$ are non-active, so each value $f(e)$ in the double sum in Eq. (13) is either at its upper bound $u(e)$ or at its lower bound $-u(e')\gamma(e')$, where e' is the arc reverse to arc e. Since arcs $e_{r,w}$ and $e_{r,w}$ do not form a contractable edge, at least one of them must have small residual capacity, that is

$$0 < u_{f,\mu}(e_{r,w}) \leqslant \Delta_{f,\mu} \quad \text{or} \quad 0 < u_{f,\mu}(e_{w,r}) \leqslant \Delta_{f,\mu}.$$

The flow value $f(e_{r,w})$ can be expressed in the following two ways:

$$f(e_{r,w}) = u(e_{r,w}) - u_f(e_{r,w}) \tag{14}$$

$$= -\gamma(e_{w,r})u(e_{w,r}) + \gamma(e_{w,r})u_f(e_{w,r}). \tag{15}$$

If $u_{f,\mu}(e_{r,w}) \leqslant \Delta_{f,\mu}$, then we substitute $f(e_{r,w})$ in (13) according to (14). Now each term on the left-hand side of (13) except $-u_f(e_{r,w})$ involves only the fractional input numbers. Therefore, by the analogous argument as in the proof of Lemma 11, the left-hand side of Eq. (13) excluding the term $-u_f(e_{r,w})$ is a fractional number with

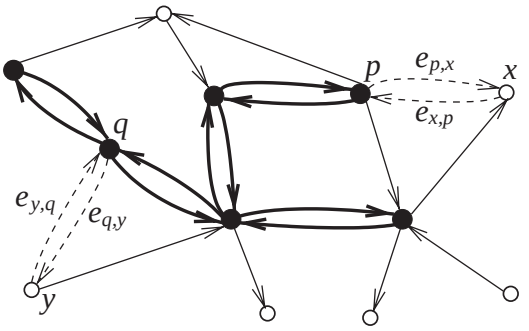

Fig. 7. Strong component considered in Lemma 13.

denominator equal to the product of at most $2\deg(C)+1 \leqslant 2d-1$ integers not greater than B. Thus, the absolute value of this number is either equal to 0 or greater than B^{-2d}. We obtain contradiction with Eq. (13) because

$$0 < u_f(e_{r,w}) = u_{f,\mu}(e_{r,w})/\mu(r) \leqslant \Delta_{f,\mu}B^{n-1} < B^{-3d+n} \leqslant B^{-2d}.$$

If $u_{f,\mu}(e_{r,w}) > \Delta_{f,\mu}$, then $u_{f,\mu}(e_{w,r}) \leqslant \Delta_{f,\mu}$. In this case, we substitute $f(e_{r,w})$ in (15) according to (15) and conclude that the absolute value of the left-hand side of Eq. (13) excluding the term $\gamma(e_{w,r})u_f(e_{w,r})$ is either equal to 0 or greater than B^{-2d}. We obtain contradiction with Eq. (13) because $\gamma(e_{w,r})u_f(e_{w,r})$ is positive and smaller than B^{-2d} (recall that $\gamma_\mu(e_{w,r}) = 1$, because arc $e_{w,r}$ is active):

$$\begin{aligned} 0 < \gamma(e_{w,r})u_f(e_{w,r}) &= [\gamma_\mu(e_{w,r})\mu(w)/\mu(r)][u_{f,\mu}(e_{w,r})/\mu(w)] \\ &= u_{f,\mu}(e_{w,r})/\mu(r) \leqslant \Delta_{f,\mu}B^{n-1} < B^{-3d+n} \leqslant B^{-2d}. \end{aligned}$$

Lemma 13. *Let $G_{f,\mu}$ be a basic canonical network and let $C \subseteq V\setminus\{t\}$ be a strong component of $G_{f,\mu}$ with the following properties. There is no residual supply in C and there are exactly two active but non-contractable pairs of reverse arcs $\langle e_{x,p}, e_{p,x}\rangle$ and $\langle e_{y,q}, e_{q,y}\rangle$ adjacent to C, where $p, q \in C$. If an integer d is such that $d \geqslant n$ and $\Delta_{f,\mu} \leqslant B^{-3d}$, then*

(a) *the degree of set C is at least d, or*

(b.1) $u_{f,\mu}(e_{p,x}) = u_{f,\mu}(e_{y,q}) \leqslant \Delta_{f,\mu}$, *or*

(b.2) $u_{f,\mu}(e_{x,p}) = u_{f,\mu}(e_{q,y}) \leqslant \Delta_{f,\mu}$.

Proof. Let T be the bidirectional tree of active arcs spanning C and let r be an arbitrary node in C. Fig. 7 is an example of a strong component considered in this lemma. Lemma 12. We assume that the degree of set C is at most $d-1$ and show that one of conditions (b.1) or (b.2) must hold. We proceed in a similar way as in

proof of Lemma 12. Let $f(e_{p,x}) = \alpha_p - \beta_p$ and $f(e_{q,y}) = \alpha_q - \beta_q$, where

$$\begin{aligned} \alpha_p &= u(e_{p,x}) \\ \beta_p &= u_f(e_{p,x}) \end{aligned} \quad \text{if} \quad 0 < u_{f,\mu}(e_{p,x}) \leqslant \Delta_{f,\mu}, \tag{16}$$

$$\begin{aligned} \alpha_p &= -\gamma(e_{x,p})u(e_{x,p}) \\ \beta_p &= -\gamma(e_{x,p})u_f(e_{x,p}) \end{aligned} \quad \text{otherwise} \tag{17}$$

and α_q and β_q are defined analogously. Take Eq. (10) for this case and substitute $f(e_{p,x})$ and $f(e_{q,y})$ with $\alpha_p - \beta_p$ and $\alpha_q - \beta_q$ to obtain the following equation:

$$-\sum_{v\in C} \delta(v)\gamma(P_v) + \sum_{v\in C} \sum_{e\in E_v^- \setminus (T\cup\{e_{p,x},e_{q,y}\})} f(e)\gamma(P_v)$$

$$+\alpha_p\gamma(P_p) + \alpha_q\gamma(P_q) - \beta_p\gamma(P_p) - \beta_q\gamma(P_q) = 0. \tag{18}$$

All terms on the left-hand side of (18) except the last two involve only the fractional input numbers. Therefore, using the same argument as in the proofs of Lemmas 11 and 12, the absolute value of the left-hand side of (18) without the last two terms must be either equal to 0 or greater than B^{-2d}. Now we bound the last two terms. If the flow value $f(e_{p,x})$ is such that case (16) applies, then using the fact that $\gamma_\mu(P_v) = 1$ for $v \in C$, we have

$$0 < \beta_p\gamma(P_p) = u_f(e_{p,x})\gamma(P_p) = [u_{f,\mu}(e_{p,x})/\mu(p)][\gamma_\mu(P_p)\mu(p)/\mu(r)]$$

$$= u_{f,\mu}(e_{p,x})/\mu(r) \leqslant \Delta_{f,\mu}B^{n-1} \leqslant B^{-3d+n-1} \leqslant B^{-2d-1}.$$

If case (17) applies, then $0 < u_{f,\mu}(e_{x,p}) \leqslant \Delta_{f,\mu}$ (edge $\{p,x\}$ is not contractable, so the residual capacities $u_{f,\mu}(e_{p,x})$ and $u_{f,\mu}(e_{x,p})$ are not both large) and we have

$$0 < -\beta_p\gamma(P_p) = \gamma(e_{x,p})u_f(e_{x,p})\gamma(P_p)$$

$$= [\gamma_\mu(e_{x,p})\mu(x)/\mu(p)][u_{f,\mu}(e_{x,p})/\mu(x)][\gamma_\mu(P_p)\mu(p)/\mu(r)]$$

$$= u_{f,\mu}(e_{x,p})/\mu(r) \leqslant B^{-2d-1}.$$

Thus $0 < |\beta_p\gamma(P_p)| \leqslant B^{-2d-1}$. Analogously for the term $\beta_q\gamma(P_q)$, $0 < |\beta_q\gamma(P_q)| \leqslant B^{-2d-1}$. Therefore, the terms $\beta_p\gamma(P_p)$ and $\beta_q\gamma(P_q)$ in (18) must cancel each other, or otherwise the left-hand side of (18) would not sum up to zero. Moreover, if case (16) applies, then we must have

$$0 < u_{f,\mu}(e_{p,x})/\mu(r) = \beta_p\gamma(P_p) = -\beta_q\gamma(P_q) = u_{f,\mu}(e_{y,q})/\mu(r)$$

and if case (17) applies, then we must have

$$0 < u_{f,\mu}(e_{x,p})/\mu(r) = -\beta_p\gamma(P_p) = \beta_q\gamma(P_q) = u_{f,\mu}(e_{q,y})/\mu(r).$$

Hence in case (16), condition (b.1) holds, in case (17) condition (b.2) holds.

Lemma 14. *Let $G_{f,\mu}$ be a canonical network. If $\Delta_{f,\mu} < B^{-(n+1)}/2$, then at least one arc in every pair of reverse arcs in $G_{f,\mu}$ has large residual capacity.*

Proof. Let $e_{v,w}$ and $e_{w,v}$ be a pair of reverse arcs and assume that $u(e_{v,w}) > 0$ (the capacity of at least one of these two arcs must be positive). The expression $u_f(e_{v,w}) + u_f(e_{w,v})$ is a linear function of the flow $f(e_{v,w})$, so its minimum value is

$$u(e_{w,v}) + \gamma(e_{v,w})u(e_{v,w}) \geqslant \gamma(e_{v,w})u(e_{v,w}) \geqslant B^{-2},$$

when the flow $f(e_{v,w})$ is at its upper bound $u(e_{v,w})$, or

$$u(e_{v,w}) + \gamma(e_{w,v})u(e_{w,v}) \geqslant u(e_{v,w}) \geqslant B^{-1},$$

when the flow $f(e_{v,w})$ is at its lower bound $-\gamma(e_{w,v})u(e_{w,v})$.

If network $G_{f,\mu}$ is canonical and $\Delta_{f,\mu} < B^{-(n+1)}/2$, then at least one of the residual capacities $u_{f,\mu}(e_{v,w})$ and $u_{f,\mu}(e_{w,v})$ must be large, that is, greater than $\Delta_{f,\mu}$, because their sum is greater than $2\Delta_{f,\mu}$:

$$\begin{aligned} u_{f,\mu}(e_{v,w}) + u_{f,\mu}(e_{w,v}) &= u_f(e_{v,w})\mu(v) + u_f(e_{w,v})\mu(w) \\ &\geqslant [u_f(e_{v,w}) + u_f(e_{w,v})]B^{-(n-1)} \\ &\geqslant B^{-2} \cdot B^{-(n-1)} > 2\Delta_{f,\mu}. \end{aligned}$$

□

6. Proof of Theorem 8

Apply procedure SHRINK to a basic canonical network $G_{f,\mu}$ such that $\Delta_{f,\mu} \leqslant B^{-3kn}$, $1 \leqslant k \leqslant m/n$, and let $G^{(1)}$, $G^{(2)}$ and $G^{(3)}$ denote the networks at the end of steps 1, 2 and 3, respectively. Call a strong component of network $G_{f,\mu}$ large, if its degree is at least kn, and small otherwise. The nodes of network $G^{(3)}$ correspond to some strong components of network $G_{f,\mu}$. Call a node of network $G^{(3)}$ large or small, depending whether it corresponds to a large or small strong component of $G_{f,\mu}$. The core of the proof is to show that network $G^{(3)}$ has $O(m/(kn))$ nodes. We show this by first showing that network $G^{(3)}$ has at least as many large nodes as the small ones, and then showing a bound on the number of large strong components of $G_{f,\mu}$.

Let C be a component of network $G_{f,\mu}$ and estimate how many small and large nodes this component contributes to the final network $G^{(3)}$. If C consists of only one strong component and contains the sink t, then it becomes the sink node in network $G^{(3)}$, and it may be either large or small. If C consists of only one strong component, does not contain the sink t, and does not have any residual supply, then it is contracted into a single node v in network $G^{(1)}$ which is free: for every pair of reverse arcs $e_{v,x}$ and $e_{x,v}$ in $G^{(1)}$, the residual capacity of one of them is zero (since these arcs do not form an active edge) and the residual capacity of the other one is large (see Lemma 14). If C consists of only one strong component but does have residual supply, then Lemma 11

implies that C is a large component, so it contributes to network $G^{(3)}$ only one node and this node is large.

For the remaining case, that is, when component C consists of at least two strong components, let W be the set of nodes of network $G^{(3)}$ contributed by C, and let T be the tree in $G^{(3)}$ of the active edges spanning W. One node in W may be the sink or a node with positive residual supply. Let $W' \subseteq W$ denote the set of the other nodes in W. Lemma 12 implies that a node in W' which is a leaf of tree T must be large. Lemma 13 implies that a node in W' of degree 2 in T must be also large (otherwise it would have been shortcut, become a free node, and then removed). Thus, each small node in W' has at least degree 3 in T. There are at most $|W|/2 - 1$ nodes in T of degree at least 3, so there are at most $|W|/2$ small nodes in W.

Let n' and n'' denote the number of nodes in $G^{(3)}$ and the number of large nodes in $G^{(3)}$, respectively. Putting together all cases considered in the previous two paragraphs, we conclude that $n'' \geqslant n'/2-1$. At least n'' strong components of network $G_{f,\mu}$ are large, so at least $knn''/2$ arcs of $G_{f,\mu}$ are adjacent to strong components. Thus, $m \geqslant knn''/2$, so $n'' \leqslant 2m/(kn)$ and $n' \leqslant 5m/(kn)$ (assuming that $n' \geqslant 10$). For each pair of nodes v and w in $G^{(3)}$, there may be at most one large capacity arc from v to w. This and Lemma 14 imply that there are at most 4 arcs between any two nodes in network $G^{(3)}$: a small-capacity arc in one direction, a parallel large-capacity arc with a smaller gain factor, and the two arcs reverse to these two. Hence, the number of arcs m' in network $G^{(3)}$ is at most $2(n')^2$, so the number of arcs in the network returned by procedure SHRINK($G_{f,\mu}$) is

$$\min\{m, m'\} \leqslant \min\{m, 2(n')^2\} \leqslant \min\left\{m, 2n^2, 2\left(\frac{5m}{kn}\right)^2\right\}.$$

7. Further questions

Tardos and Wayne [14] proposed simple combinatorial algorithms for the maximum generalised flow problem, including a generalisation of Goldberg and Tarjan's push-relabel algorithm for the min-cost flow problem [5]. Can our network-modification operations improve the time bounds of those algorithms by factor m/n? Wayne [16] showed a polynomial-time combinatorial algorithm for the *minimum cost* generalised flow problem. As in the maximum generalised flow algorithms, the computation of Wayne's algorithm ends when the value of flow is within $B^{-\Omega(m)}$ from optimal. Can some network-modification operations work for this algorithm?

Polynomial bounds for linear programming problems are probably more commonly expressed not in terms of parameter B but in terms of the parameter $L = \sum\{\lceil \log(i+1) \rceil : i \in D_{\text{int}}\}$, where D_{int} is the set of the enumerators and denominators of the fractional input numbers (that is, L is the binary length of the input). Since $L = O(m \log B)$, we can always substitute in upper time bounds term L with $m \log B$, but term $\log B$ cannot be substituted with L/m without further argument. In Theorem 2, the B^{-m} bound on the residual supply can be replaced with $2^{-O(L)}$ (see the proofs of this theorem in [4,7]). This implies that algorithm $\mathscr{A}$ of Section 3 computes

near-optimal flows in $O(L)$ iterations and the running times of Goldfarb et al's algorithms are $\tilde{O}(m^2L)$. Can our analysis be extended to show that maximum generalised flows can be computed in $\tilde{O}(nmL)$ time?

And finally, do the network-modification operations presented in this paper bring us any closer to settling the big open question of computing maximum generalised flows in strongly polynomial time?

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Engle Wood Chiffs, NJ, 1993.
[2] L.R. Ford Jr., D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
[3] F. Glover, J. Hultz, D. Klingman, J. Stutz, Generalized networks: a fundamental computer-based planning tool, Manage. Sci. 24 (12) (1978) 1209–1220.
[4] A.V. Goldberg, S.A. Plotkin, E. Tardos, Combinatorial algorithms for the generalized circulation problem, Math. Oper. Res. 16 (2) (1991) 351–381.
[5] A.V. Goldberg, R.E. Tarjan, Finding minimum-cost circulations by successive approximation, Math. Oper. Res. 15 (1990) 430–466.
[6] D. Goldfarb, Z. Jin, A faster combinatorial algorithm for the generalized circulation problem, Math. Oper. Res. 21 (1996) 529–539.
[7] D. Goldfarb, Z. Jin, J. Orlin, Polynomial-time highest-gain augmenting path algorithms for the generalized circulation problem, Math. Oper. Res. 22 (1997) 793–802.
[8] A. Kamath, O. Palmon, Improved interior point algorithms for exact and approximate solution of multicommodity flow problems, in: Proc. 6th Annu. ACM-SIAM Symp. on Discrete Algorithms, San Francisco, California, 1995, pp. 502–511.
[9] S. Kapoor, P.M. Vaidya, Speeding up Karmarkar's algorithm for multicommodity flows, Math. Programming 73 (1) (1996) 111–127.
[10] S.M. Murray, An interior point approach to the generalized flow problem with costs and related problems, Ph.D. Thesis, Stanford University, August 1992.
[11] K. Onaga, Optimal flows in general communication networks, J. Franklin Inst. 283.
[12] T. Radzik, Faster algorithms for the generalized network flow problem, Math. Oper. Res. 23 (1) (1998) 69–100.
[13] D.D. Sleator, R.E. Tarjan, Self-adjusting binary search trees, J. Assoc. Comput. Mach. 32 (1985) 652–686.
[14] E. Tardos, K. Wayne, Simple generalized maximum flow algorithms, in: Proc. 6th Internat. Conf. on Integer Programming and Combinatorial Optimization, Houston, Texas, 1998, pp. 310–324.
[15] P. M. Vaidya, Speeding up linear programming using fast matrix multiplication, in: Proc. 30th IEEE Annu. Symp. on Foundations of Computer Science, Research Triangle Park, North Carolina, 1989, pp. 332–337.
[16] K. Wayne, A polynomial combinatorial algorithm for generalized minimum cost flow, in: Proc. 31st Annu. ACM Symp. on Theory of Computing, Atlanta, Georgia, 1999.